FlexJS

Digging deeper

OmPrakash Muppirala

About me

- OmPrakash Muppirala
- Lead UI Developer at Machine Zone
- Member@ASF
- Apache Flex Committer and PMC Member
- UI Architect with 15+ years experience building UI for enterprise and consumer apps
- @bigosmallm

What is FlexJS?

- A new application development framework
- Cross compiles MXML and ActionScript into HTML and JavaScript
- Runs in web, desktop and mobile(app) in swf format
- Runs in browser based environments as HTML/JS/CSS files **without** the need for the Flash Player plugin.
- Brings the advantages of Flex to the JavaScript world

Highlights

- MXML DataBinding
- MXML States
- Classes, interfaces using ActionScript 3.0
- Component Library
  - Button, Label, DropDownList, Panel, Charts, Map, etc.
- Bring your favorite design patterns (Singleton, MVC, MVVM, etc.)
- Unit test with FlexUnit
- Metadata to annotate classes

Highlights

- New coding patterns support plug-ins and composition.
  - Strands and Beads
  - Incremental feature development
  - Better Performance
  - Smaller SWFs and JS downloads
  - Pay as you go

Highlights

- Ease of use and robust application development
  - Full fledged object oriented inheritance model (not just prototype based)
  - Statically type code, dynamic optional. So best of both worlds.
  - Packages, classes, interfaces
  - Use standard design patterns easily
  - Easy to maintain
  - Friendly for projects with medium to large size teams

Declarative layout, scripted behavior

MXML:

```
<js:TextButton id="refreshButton" text="Refresh" x="300" y="10" click="refreshBtnClick(event)"/>
```

ActionScript3:

```
private function refreshBtnClick(event:Event):void
{
    var e:Event = new Event("refresh",true);
    dispatchEvent(e);
}
```

Reusable components – the holy grail

- AngularJS, ReactJS, WebComponents/Polymer all try to achieve the concept of reusable components
- Each MXML file is a class by itself and is 100% reusable
- Is-a and Has-a relationships
- No need for templates, no need to force users to follow a specific pattern
- Set properties as attributes and nodes in MXML, not as unreadable complex objects

The beauty is in the details

AngularJS Directive:

HTML:

```
<highchart id="chart1" config="chartConfig"></highchart>
```

JavaScript:

```
var chartConfig =
{ options:
    {
      chart: { type: 'bar' },
      tooltip: { style: { padding: 10, fontWeight: 'bold' }
    },
series: [
    {
        data: [10, 15, 12, 8, 7] }],
        title: { text: 'Hello' },
        loading: false,
        xAxis: { currentMin: 0, currentMax: 20, title: {text: 'values'}
    },
    useHighStocks: false,
    size: { width: 400, height: 300 },
func: function (chart) { //setup some logic for the chart } };
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                 <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                  <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
        </fx:Component>
      </js:itemRenderer>
    </js:ColumnSeries>
  </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                 <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

The beauty is in the details

MXML:

```
<js:ColumnChart id="columnChart" >
  <js:beads>
    <js:SimpleBinding sourceID=“model” sourcePropertyName=“obj”
          destinationPropertyName=“dataProvider” />
    <js:HorizontalCategoryAxisBead categoryField="date" />
    <js:VerticalLinearAxisBead valueField="users" />
  </js:beads>
  <js:series>
    <js:ColumnSeries yField="users">
      <js:itemRenderer>
        <fx:Component>
          <js:BoxItemRenderer>
            <js:fill>
                <js:SolidColor color="#7CB5EC" alpha="1.0" />
            </js:fill>
          </js:BoxItemRenderer>
         </fx:Component>
        </js:itemRenderer>
      </js:ColumnSeries>
    </js:series>
</js:ColumnChart>
```

Multiple runtime targets

- Consistent behavior, visual rendering across all supported platforms
- HTML5/SVG/JavaScript
- Flash
- Mobile (via Adobe AIR or PhoneGap)
- Anywhere HTML runs!

Layout

Vertical Layout

```
<js:Container x="10" y="10" >
    <js:beads>
        <js:VerticalLayout />
    </js:beads>
    <js:Label text=“My label" />
    <js:TextButton text=“Click" />
</js:Container >
```

Horizontal Layout

```
<js:Container x="10" y="10" >
    <js:beads>
        <js:HorizontalLayout />
    </js:beads>
    <js:Label text=“My label" />
    <js:TextButton text=“Click" />
</js:Container >
```

Behavior

Inline:

```
<js:TextButton id="refreshButton" text="Refresh“
    click=“dispatchEvent(new Event(“refresh”,true))"/>
```

Function Call:

```
<js:TextButton id="refreshButton" text="Refresh“ click=“handleClick(event)"/>

<fx:Script>
    <![CDATA[
        private function refreshBtnClick(event:Event):void
            {
                var e:Event = new Event("refresh",true);
                dispatchEvent(e);
            }
    ]]>
</fx:Script>
```

Data Binding

Inline Binding :

```
<js:Label id="msgLbl" text="{this.labelText}" width="500"  />
<js:TextButton id="msgLbl" text=“Load" click=“refreshBtnClick(event)”  />

Elsewhere in the component:
<fx:Script>
   <![CDATA[

    [Bindable] public var labelText:String = “Last 30 days";

    private function refreshBtnClick(event:Event):void
    {
        labelText = "Loading last 30 days";
        var e:Event = new Event("refresh",true);
        dispatchEvent(e);
    }
  ]]>
</fx:Script>
```

Data Binding

Event based binding:

```
<js:SimpleBinding
    sourceID="applicationModel"
    sourcePropertyName="lastThirtyDaysUsers“
    destinationPropertyName="dataProvider“
    eventName="lastThirtyDaysDataChanged“ />
```

ApplicationModel.as

```
<snip>
[Bindable("lastThirtyDaysDataChanged")]
public function get lastThirtyDaysUsers():Array
{
    return _lastThirtyDaysUsers;
}

public function set lastThirtyDaysUsers(v:Array):void
{
    if(v != _lastThirtyDaysUsers)
        {
            _lastThirtyDaysUsers = v;
            dispatchEvent(new Event("lastThirtyDaysDataChanged"));
        }
}
</snip>
```

States

Define states:

```
<js:states>
    <js:State name="summary" />
    <js:State name="details" />
</js:states>
```

In MXML, define behavior based on currentState:

```
<js:Label id="titleLbl" width="300"
    text.summary="Summary" text.details="Details"/>
<js:TextArea width="300"
    height.summary="25" height.details="400"
    text.summary="{model.summaryStr}" text.details="model.detailsStr" />
<js:TextButton text="Export Details" click="exportDetails(event)"
    includeIn="details" />
```

In ActionScript, set the current state:

```
private function showSummary():void
{
    currentState = "summary";
}
private function showDetails():void
{
    currentState = "details";
}
```

Effects

```
<js:states>
        <js:State name="HomeState" />
        <js:State name=“ProductsState" />
</js:states>

<js:Container width="990" id="viewholder">
  <HomeView id="homeView" width="100%" height="550" includeIn="HomeState“ />
  <ProductsView id="pView" includeIn="ProductsState" width="100" height="550" />
</js:Container>

<js:transitions>
    <js:Transition fromState="HomeState" toState=“ProductsState">
        <js:Wipe direction="up" target="homeView" />
    </js:Transition>
    <js:Transition fromState=“ProductView" toState="HomeState">
        <js:Wipe direction="down" target=“productView" />
    </js:Transition>
</js:transitions>
```

Demo

Flex Trader

JS Version

http://bigosmallm.github.io/flexjs/flextrader/js/index.html

Flash Version

http://bigosmallm.github.io/flexjs/flextrader/swf/FlexJSStore.html

Source

https://github.com/apache/flex-asjs/tree/develop/examples/flexjs/FlexJSStore

New – manipulate HTML DOM via ActionScript3

- New experimental functionality
- Lets you manipulate DOM elements directly via ActionScript3
- All the object oriented goodness of AS3
- Code completion of HTML DOM properties
- More work happening on this as we speak

Demo

US States Map

Demo

http://bigosmallm.github.io/flexjs/mapexample/

Source

https://github.com/apache/flex-asjs/tree/develop/examples/native/USStatesMap/

Volunteers needed!

- Testing
- Development
- Documentation
- Writing Example Applications
- Any other way you can help!

Questions?

Wiki: http://s.apache.org/flexjs-wiki

Mailing List: dev@flex.apache.org

Twitter: @bigosmallm